

# 图形学实验 PA0：光栅图形学

指导教师：胡事民 助教：曹耕晨、彭浩洋

2023 年 3 月 12 日

## 1 实验综述

图形显示器可以看做一个像素的矩阵，光栅图形学（Raster Graphics）为我们提供了在像素矩阵上进行绘制的一系列方法。本次实验要求同学们自学《计算机图形学基础教程（第 2 版）》第 2 章的相关内容，编程实现直线绘制、圆形绘制以及区域填充算法。

## 2 细节要求

### 2.1 直线段的扫描转换

在解析几何中直线的方程可以表示为  $y = kx + b$ ，为了将连续的线“离散化”到像素矩阵网格（即计算机屏幕）中，我们采用光栅图形学中采用最广泛的 Bresenham 直线扫描转换算法。对于斜率  $0 \leq k \leq 1$  的直线而言，我们循环起点到终点的  $x$  列像素坐标  $x_i$ ，依次计算对应  $y_i$  的坐标。每当  $x_i$  增加一个像素的时候， $y_i$  要么保持不变，要么也增加一个像素。是否增 1 取决于误差项  $d$  的值。误差项  $d$  的初值  $d_0 = 0$ ， $x_i$  每增加 1， $d$  的值就要增加  $k$ ，当  $d \geq 0.5$  时， $y_i$  就要增 1，同时误差项  $d$  要减 1<sup>1</sup>。具体图例和伪代码请参见图形学课本 2.1.3 节。在实现的时候请注意处理各种直线斜率  $k$  的特殊情况。

### 2.2 圆的扫描转换

一个圆心为  $(x_c, y_c)$ ，半径为  $r$  的圆的隐式表达式为  $(x - x_c)^2 + (y - y_c)^2 = r^2$ 。由于圆形具有高度的对称性，我们将其分成 8 份，因此只需要扫描转换 1/8 的圆弧，就能够利用对称性绘制出整个圆形。请参见图形学课本的 2.2.2 节进行代码实现。

### 2.3 区域填充

在 Windows 的“画图”软件<sup>2</sup>中，有一个工具名为“油漆桶”，当用户给定一个种子点之后，该种子点周围相同颜色的像素都会被染成新的颜色，这种填充方式叫做“漫水填充”（Flood Fill）。这种技术实际上是通过宽度优先遍历实现的，通过一个遍历队列，就能够选取到所有符合条件的待染色点。请参见图形学课本的 2.3.2 节进行代码实现，我们推荐使用非递归版本的实现，因为实际操作的图像可能会很大。

<sup>1</sup>在计算增量时可以用类似“通分”的方式回避浮点数运算

<sup>2</sup>使用 Linux 的同学可以搜索一款 KolourPaint 的软件

## 3 框架代码说明

### 3.1 环境配置与编译

环境要求：

- Windows: WSL 下安装 CMake
- Linux: CMake
- MacOS: CMake

如果你使用其他系统的时候遇到了编译问题，请先尝试自行解决。我们的框架代码没有任何外部依赖，请在包含有 `run_all.sh` 的文件夹下打开终端，并执行：

```
1 bash ./run_all.sh
```

这段脚本会自动设置编译，并在 2 个测例上运行你的程序。你的程序最终会被编译到 `bin/PA0` 中，而输出的图片位置在 `output/` 文件夹中。我们在框架代码中去除了一些核心逻辑，因此现在这段代码仅仅是能编译而已。

### 3.2 代码结构

参考 `src/main.cpp` 文件：本程序会首先使用 `CanvasParser` 读入配置文件，该配置文件定义了需要绘制的形状集合。接着程序会按照顺序遍历所有的形状元素 (`Element`)，依次执行他们的 `draw(Image&)` 方法，最终会将绘制完成的图像存储成 `bmp` 格式。

`CanvasParser` 类负责配置文件的解析，已经为你实现好，无须更改。但建议进行阅读，了解配置文件的文件格式。

`Image` 类负责图像的读写，如下示例代码将创建一个大小为  $10 \times 15$  的图像，并在 (5,5) 像素的位置写入纯红色：

```
1 Vector3f pixelColor(1.0f, 0, 0);
2 // 10 is width, 15 is height.
3 Image image(10, 15);
4 image.SetPixel(5, 5, pixelColor);
5 // Output will be saved to demo.bmp file.
6 image.SaveImage("demo.bmp");
```

`Element` 是所有绘制操作的基类，其继承者包括线绘制 `Line`，圆绘制 `Circle` 和区域填充 `Fill`，本次编程作业中你需要实现的就是他们的 `draw` 方法。

## 4 测试用例

为了测试代码是否正确无误，我们构建了 2 个测试用例，你也可以根据文件格式构造样例进行自我测试。我们在检查作业的时候有可能会加入其他测试样例，请注意代码的鲁棒性。

**基础测试** `testcases/canvas01_basic.txt` 包含了 6 个线段和 3 个圆形，参考效果如图1所示。

**笑脸表情** `testcases/canvas02_emoji.txt` 参照 emoji 表情绘制了一个笑脸图案。绘制半圆的时候，先绘制一个整圆，再从中间用直线进行截断，只对半部分就行区域填充即可。参考图如图2所示。

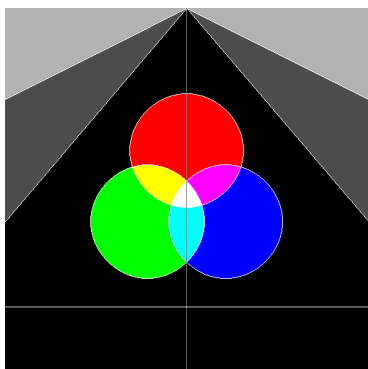


图 1: 基础测试



图 2: 笑脸表情

## 5 作业要求

以下要求假设你的当前目录为 `code`。

- 在 `include/element.hpp` 中实现 `Line`、`Circle` 与 `Fill` 的 `draw` 方法。
- 确认可以在修改且只修改 `include/element.hpp` 后，执行 `bash ./run_all.sh` 能够在 `output` 文件夹下输出两张正确的测例。
- 撰写报告，报告中应包含且只包含以下几个部分：
  1. 代码逻辑：实现时用到的画线、画圆和区域填充逻辑。
  2. 代码参考：完成作业的时候和哪些同学进行了怎样的讨论？是否借鉴了网上/别的同学的代码？
  3. (可选) 问题：你在实现过程中遇到了哪些问题？
  4. (可选) 未解决的困难：你的代码有哪些未解决的 bug？如果给你更多时间来完成作业，你将会怎样进行调试？
  5. (可选) 建议：你对本次作业有什么建议？文档或代码中有哪些需要我们改进的地方？
- 将 `include/element.hpp` 与你的报告 `README.pdf` 打包成一个 `zip` 文件提交到网络学堂。

具体文件树结构如下所示，不遵循该结构的提交会被扣分：

```
姓名-学号-PA0.zip
├── README.pdf
└── element.hpp
```

本次作业的 Deadline 以网络学堂为准。迟交的同学将得到一定的惩罚：晚交 3 天内分数将降低为 80%，3 天以上 1 周以内分数降为 50%，迟交一周以上的同学分数为 0。

## 6 致谢

本实验代码有部分借鉴于 MIT Open Courseware，按照其发布协议，本文档原则上允许同学们以 CC BY-NC-SA 4.0 协议共享引用，但是由于教学需要请同学们尽量不要将本文档或框架代码随意传播，感谢同学们的支持。